Live Play Prediction in Canadian University Football

By

Jay Turnsek

Submitted in partial fulfillment of the
requirements for the degree of
Bachelor of Science with Honours in Computer Science
at
Saint Francis Xavier University
Antigonish, Nova Scotia
May 2024

Saint Francis Xavier University
Department of
Mathematics, Statistics and Computer Science

The undersigned hereby certify that they have read a thesis entitled "**Live Play Prediction in Canadian University Football**" by **Jay Turnsek** in partial fulfillment of the requirements for the degree of **Bachelor of Science with Honours**.

Dated: _____

Supervisor: _____
Dr. James Hughes

Second Reader: _____
Dr. Taylor Smith

St. Francis Xavier University

May 2024

Author:	Jay Turnsek
Title:	Live Play Prediction in Canadian University Football
Department:	Mathematics, Statistics and Computer Science
Faculty:	Science
Convocation	May 2024

This research is dedicated to my friends and coaches, who hyped me up at every step along the way; and my mom for supporting me throughout everything.

Abstract

This paper explores the viability of predictive models for live in-game play type prediction in Canadian university football and evaluates their utility beyond traditional large-scale prediction experiments. Employing a methodology that involves data gathering from a university football season, feature annotation, engineering, hyperparameter optimization, and model testing through k-fold validation, our research achieved a notable 79.8% accuracy on the final game trial. Remarkably, this accuracy surpassed existing studies with larger datasets, demonstrating the effectiveness of our approach with a limited dataset of 820 samples.

Analysis of feature importance across models revealed nuanced performance variations in predicting play types, with some models excelling in obvious scenarios and others thriving in less predictable situations. Furthermore, our research sheds light on the challenges associated with predicting coordinators' play calls and highlights the nonlinear relationships between dataset variables that posed difficulties for the models.

This study not only advances the field of sports analytics and machine learning but also provides a foundation for further exploration of live in-game prediction. Our research opens the door to tools that empower coaches to make informed decisions during games, ultimately contributing to enhanced team performance and increased win percentages. The target audience includes researchers and practitioners in sports

analytics and machine learning, as well as coaches and football staff. This work signifies a crucial step forward in understanding the complexities of live play prediction, showcasing its potential impact on strategic decision-making in sports.

Acknowledgements

This study would not have been possible without the guidance and overseeing of Dr. James Hughes, the supervisor for this thesis. I am extremely grateful for his consistent optimism and honesty at each step of the way. I would also like to thank Dr. Taylor Smith for taking the time to read the thesis, and Dr. Iker Gondra for providing the opportunity to complete this research.

I would also like to express my gratitude to my coaches, Gary Waterman and Jon Svec for giving me something passionate to write about; as well as supply the data and give great context and input throughout the process.

Finally, I'd like to thank my family and close friends for the continuous encouragement throughout the writing process, and listening to me babble on about what I was working on.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Canadian football is played at all levels in the country, from youth leagues all the way up to professional leagues like the CFL. Closely related to American football, with the exception of some rugby rules being retained today and notable changes like the size of the field, the sport has seen continued growth since the first national championship in 1909 [5]. Canadian football occurs on a play-to-play basis, where individual plays are akin to chess moves. One team is on offence and the other is on defence, where the main goal is for the offence to advance the ball to their opponent's end zone; conversely, the defence aims to stop the offence from advancing the ball. On any given play, the offence has a set play call where players align and complete responsibilities in a structured manner. The main two play types are "run" and "pass". In a run play, the offence hands the ball to a ball carrier after the snap and other offensive players block for the ball carrier. On pass plays, the quarterback attempts to throw the ball to a receiver while they run downfield, typically on a specific path called a "route". These structured play calls have

been the paramount concern for the preparation of defences, who typically watch video from previous games to identify patterns in the offensive playcalling. Recently, preparation tactics have advanced into software solutions with tools like *HUDL/DVSport* and *GameStrat* to analyze past and current games, respectively.

1.2 Motivation

Defensive playcalls aim to anticipate what the offence is trying to do; obviously, if you know an offence is trying to pass the ball, you would call plays that assign more players to pass coverage. In Canadian university football, players are able to watch game film to identify specific *player* tendencies, where coaches/coordinators have a less granular approach focusing on *playcall* tendency.

The analysis of these tendencies is only as complex as a decision tree with a depth of 2, as shown in Figure 1.1, and there currently exists no solution to put more powerful analytics in the hands of the coaches in a live-game setting. For example, given a down of 1 and distance of 10, we could be presented with 47% run, and 53% pass for a specific team. This is very limited as complex relationships in the data are not captured. With more predictive power, coaches could make the optimal play call on each play given the current situation, leading to improved anticipation and performance of the defence, which then leads to the opponent scoring fewer points and more wins logged. There have been numerous studies on predicting run or pass plays in the NFL, notably by Joash [8] and Heiny [7]. The problem in the case of this study lies in live play data availability. These papers make use of a plethora of features, most of which are unrealistic or unattainable all together during a game. In this study, we aim to find a solution to this problem.

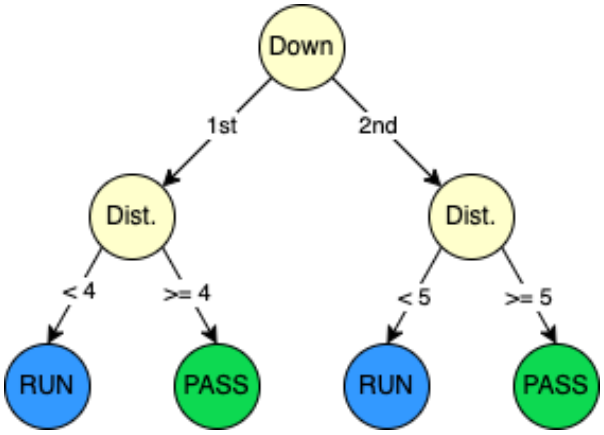


Figure 1.1: A simple decision of play call tendency.

1.3 Objectives

The goal of this research is to present a novel method for classifying offensive play calls based on the live in-game data available in Canadian university football; That is, allowing for data input, prediction, and relay of information in the 20 seconds between the spot and snap of the ball. The questions to be answered are (a) if predictive models are useful in the context of this problem, specifically on the limited set of features provided; (b) which of these features are relevant for accurate classification; and (c) in what ways we can engineer new features from the data to aid in prediction. By achieving these goals, coaches will be able to use these models from the booth to relay play type probabilities to coaches on the sidelines. Coaches can use this information to inform their play calling decisions, putting their players in better position to perform at a higher level.

1.4 Scope & Limitations

The scope of this study is solely on the offensive plays of teams within the Atlantic University Sports (AUS) football conference, and predicting one of two play types: run

or pass. The decision to limit this study to a binary classification is based on the complexity and nuances prevalent in predicting each individual offence's tendencies, with future efforts aimed at categorical classification of pass patterns and run plays. The limitations of this study as mentioned are the limited features and samples of the dataset. The short length of football seasons lead to about 600 samples per year, and the live observable features are limited to 5. Further, the process of inputting play data, getting a prediction, and relaying that message all must occur within a very short timeframe. This places a limit on model complexity and computational expense as the entire process must happen in under 20 seconds. This study is bound to the play-by-play data of the 2023 AUS Football season, making use of data from the five universities within the conference (StFX University, Acadia University, Mount Allison University, Bishops University, and Saint Mary's University). We also enforce that models can only make use of the observable features, along with the engineered features derived from them.

1.5 Methodology Overview

At a high level, the methodology of this study is as follows:

- *Data Engineering:* The engineered features are created as a function of the annotated features provided. Further, they are created using the context of football to exploit potential patterns in play calling. For example, the feature *"PREV_PLAY_RESULT"*, which denotes what happened on the previous play, including, but are not limited to, incomplete pass, rushing 1st down, etc.

- *Model Selection:*

- *Dummy Classifier:* A dummy classifier that simply always predicts the majority class was implemented to provide a simple baseline to compare other models to.
- *Decision Tree:* A decision tree based upon existing HUDL analytics was built, that takes into account simple tendency from a down and distance perspective. The choice to include this model was a secondary baseline so other models could be compared to a hypothetical implementation of existing methods used today.
- *K-Nearest Neighbours:* A simple classification algorithm that takes into account spatial relationships. This model was chosen based on the ability to interpret those relationships, as well as simplicity and ability to work with non-parametric data.
- *Logistic Regression:* A basic logistic regression was selected based on its straightforward interpretability with respect to feature importance, as well as its computational efficiency with respect to live prediction.
- *XGBoost:* XGBoost is an ensemble method that was selected based on its well-documented performance in classification tasks in numerous domains. The model typically outperforms most if not all models in classification tasks with tabular data. Further, much like logistic regression, feature importance is easily extractable from the model for analysis.
- *Neural Network:* A neural network was implemented using the Python library PyTorch. This model was selected based on its feature learning abilities and complex pattern recognition.

- *Model Training & Optimization:* Models are trained using both k-fold and application specific methods, outlined in the Chapter 5. Optimization makes use of Optuna, a hyperparameter optimization library in Python, to find the ideal parameters to maximize classification accuracy on the test data.

- *Model Evaluation:* Models are then evaluated based on accuracy in each test, and their optimization processes looked at in depth. Further, feature importance is extracted on a model-to-model basis to find out which features additional engineered features should be derived from.

1.6 Results Overview

At a high level, the results of this study are presented as follows:

- *Model Optimization:* Firstly, the optimization process of each model was analysed to check how hyperparameter changes affected results. Further, the hyperparameters gathered from this process are displayed providing transparency about the models used for this study.

- *K-Fold Results:* The results of training and testing the models using k-fold analysis is discussed in depth. Models are compared to the baseline, as well as the changes observed from the introduction of engineered features. A more in-depth analysis is made as well in relation to down and distance, as this is the typical partitioning in game preparation by coaches and coordinators.

- *Application Results:* The results of testing conducted over the course of a hypothetical season are explained here. Along with comparisons made in the k-Fold

Results section, the accuracies as the season progresses are explored, both by model and as a comparison between including and excluding engineered features.

- *Feature Importance:* Next, the importance of all the features included are displayed. The discussion relates these importances to the performances in each model, and the fluctuations between down and distance partitions. The differences in importance between models is also explored, along with SHAP values and decision boundaries of both simple and complex relationships between features.

1.7 Thesis Structure

- *Chapter 1: Introduction*: This chapter is a quick look into the contents of this thesis, giving the reader the background information needed to contextualize the chapters to follow. The overall goal is outlined, along with the constraints and scope of the paper.

- *Chapter 2: Literature Review*: This chapter is concerned with exploring works related to this study. Papers from within this domain of this study and adjacent domains are explained, with an emphasis on the capabilities of machine learning in this space, as well as the lack of research done for live game data.

- *Chapter 3: Problem Description*: Here we dive into what exactly it is we're trying to solve, as well as the limiting factors that contribute to this study. The aim is for the reader to fully understand why accuracies are less than expected, and the difficulties raised by actual application of these models.

- *Chapter 4: Dataset*: This chapter explains in depth each of the features used in this

study, providing context and explanations for football specific features. Further, the composition of engineered features with respect to the observed (annotated) features. The goal of this chapter is for the reader to fully understand each feature's definition and its importance, without any prerequisite football knowledge.

- *Chapter 5: Methodology*: In this chapter, the full implementation process is explained in depth, along with model selection decisions and their reasoning. The aim here is to explain the "why" along with the "what" of what was done in this study, and provide background for reading the "Results and Discussion" section.

- *Chapter 6: Results and Discussion*: Depicted here is how each model performed, along with extensive analysis of the optimization process and feature importance. The goal is to answer the questions we asked in our hypothesis, present a novel solution, as well as set up discussions for future development.

- *Chapter 7: Conclusions and Future Work*: Finally, we give a concise recap on the study, along with its implications. The future development discussion from the previous chapter is expanded upon, hopefully sparking ideas for future studies in this everchanging domain.

Chapter 2

Literature Review

Many studies have been conducted relating to the prediction of various factors among different sports. Some of these include pitch prediction in baseball, play type prediction in American football, and even win rate prediction in tennis relative to sportsbetting odds. Most predictive models used in related studies found success using a large dataset, but this raises a problem for real-world applications. Most football teams at all levels play between 6 to 15 games each season and therefore a model that achieves success on this smaller set of data is required to be useful on the sideline. This is the main barrier we are attempting to overcome in this study. Our dataset is from the Canadian university football season, which has only eight games in the regular season.

2.1 Existing Tools for Players and Coaches

Current tools for scouting and predicting the play calling of opposing teams do exist in the form of paid software solutions: primarily,*Hudl* and *DVSport*. These systems allow teams to compile recorded video of practice and game video, with a diverse set

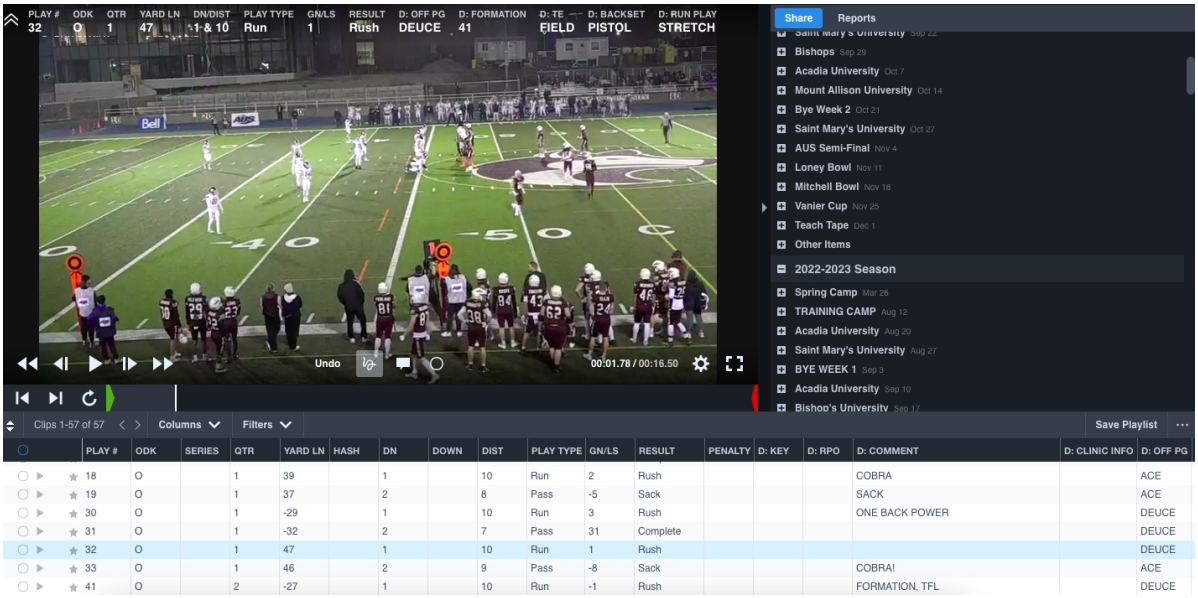


Figure 2.1: Hudl user interface.

of playback instructions to assist in coaching and technique correction. These playlists of play-by-play clips are annotated manually by the user, providing a basis for their scouting reports.

These scouting reports are then compiled by Hudl or DVSport, and display a basic table of frequency of plays based on some known variable, shown in Figure 2.2. For example, team A may call a run play 75% of the time when it is 2nd down.

The typical set of features included with one of these solutions is shown in Figure 2.1. A user is able to navigate playlists of specific games or teams, and then watch them sequentially. Annotation can be added to each clip from the game in the table view along the bottom, where features like down, distance, and formation are logged play-by-play. The playback window shows selected data from the table below and allows for precise playback controls, to pick out minute details in play and potential keys in player posture and movement.

Down/Distance vs. Run/Pass (?)

	Run	Pass
1st & 10+	48% 15/31	52% 16/31
1st & < 10	-	-
2nd & 7+	15% 3/20	85% 17/20
2nd & 4 - 7	-	100% 4/4
2nd & < 4	50% 1/2	50% 1/2
3rd & 7+	-	-
3rd & 4 - 7	-	-
3rd & < 4	-	-
4th & 2+	-	-
4th & ≤ 2	-	-

Field Position vs. Run/Pass (?)

	Run	Pass
GOLD	-	-
Blue	-	100% 2/2
Green	33% 5/15	67% 10/15
Yellow	31% 8/26	69% 18/26
Black	43% 6/14	57% 8/14

Down vs. Run/Pass (?)

	Run	Pass
1st	48% 15/31	52% 16/31
2nd	15% 4/26	85% 22/26
3rd	-	-
4th	-	-

Figure 2.2: Hudl's generated reports for a given team.

Where these tools fall short, however, is more complex indicators. The scouting reports only consider one variable and a frequency, which is short-sighted. For example, if a team is presented with a 2nd down play, they could make the prediction that it's going to be a pass by the 85% value in the Down vs. Run/Pass table. However none of these tables have a memory or temporal sense; meaning that if a run play has worked in this specific situation several times this play, the opposing coach will of course call a run play more than likely. This goes against the 85% prediction provided by Hudl, and shows that it is flawed when compared to a method that takes into account play history and specific situations. A more comprehensive approach would be to instead consider multiple or engineered features, previous plays, or any other methods used by predictive models outlined in this thesis. Aided with these additional practices, the coach could predict the run play and call a defensive play designed to stop a run play instead of a pass play, improving the probability of getting a stop on that drive.

2.2 Related Work in American Football

Previous studies have been conducted to predict a run or pass play using Markov models, achieving an impressive 71.5% accuracy [12]. Although the time series analysis used is a useful tool backed by the generalized matching law[1] in sports [16], the large dataset makes it hard to justify applicable use by teams, given that collegiate seasons are only 8 to 16 games long.

The need for interpretation in these models is paramount, as indicative rules must be memorized by defensive players and coaches at game time to identify them in a high stress environment. Typically, most advanced models like neural networks are a black box, with no indication of the rules the model created. This was explored by Joash [8], where an easy-to-memorize model achieved 86% of the accuracy that a neural network was able to when predicting run versus pass. For example, if a linebacker was able to memorize a few of the rules from the model, they could play the run aggressively from the snap, giving a step or two advantage on blockers on the offensive side of the ball.

It's possible to also garner success in more granular predictions, where a discriminant analysis approach was used to predict short/medium/long pass, run, and scramble plays [7]. This additional step in classification provides further insight for coaches, as different coverages leverage players to help on different receiver route depths. For example, a Cover 4 deep zone is more geared towards stopping a long pass, and a Cover 3 deep zone is more useful for short passing and run plays.

A study identifying receiver routes throughout an offensive play based on their paths

[1]The matching law is a behavioral principle that states that behavior occurs in direct proportion to reinforcement available for each behavior; ie, if a run play has led to success previously in this situation, it's likely to be called again. The gain of yards then would be the reinforcement.

showed promising results [9]. This study used Gaussian process regression and an additional linear model to the dynamic environment of the receiver's trajectory relative to a defender. This suggests that there could be merit in using similar methods to identify run/pass plays based on linemen stance/movement, or quarterback/running back mesh.

In all of the aforementioned studies, there has been no attempt to predict these things in a live, in-game setting; instead, they follow the classic sequence of classifying data based on a training set and a test set. In this study specifically, we are trying to adapt these practices to perform as well as possible during this live setting. These predictions are needed on a play-by-play basis, and have less value when observed in retrospect.

2.3 Related Work in Other Sports

Baseball pitches were predicted as either fastball or non-fastball in a binary classification study utilizing support vector machines (SVM) and k-nearest neighbour (KNN) algorithms [6]. A strong focus of this study was feature selection and engineering, providing an extensive and supported basis to make predictions on, for example a complex engineered feature like "Cartesian quadrant average for previous 3 pitches in specific count". This was shown to improve predictions greatly, and even when the study narrowed its features only to those known at the time of the current pitch moderate success was attained: around 75%.

Other studies from a more broad scope used similar algorithms to predict win rate for tennis players against sportsbetting odds [20], which focuses on the historical data of players and context of the current game. This proved useful for the prediction, garnering 70% average accuracy, and validates the use of historical data used in our study in conjunction with more real-time data analysis.

Chapter 3

Problem Description

3.1 Motivation

The most important skill a defensive player or coach can have is predicting what play the offence is going to run. In situations where a team is obviously more likely to pass, coaches call plays that are structured to stop a passing play specifically — for example, a pass coverage play with extra defensive backs, or a blitz with the goal of pressuring the quarterback. In situations where a team must run, the same principle applies: more defensive players are given a run responsibility from the play call.

It follows, then, that the ability to know which of the two types of plays will be called in less obvious situations would be incredibly beneficial.

If one is given an indicator prior to the play by way of a machine learning model, the message of this indicator could be relayed from the booth to the coach and then to the players in the form of the corresponding play call.

In other words, the aim is to use past and current game data to predict the play type of the next play and to aid defensive play calling, gaining a massive competitive

advantage.

This seemingly simple classification problem has many limitations. These limitations include data constraints in both features and samples, as well as a tight time constraint between prediction and the start of the next play.

3.2 Limiting Factors

3.2.1 Data Sparsity

University football teams play a very limited schedule (8-16 games) when compared to other sports like basketball or baseball, which have 35 and 56, respectively. Further, not all plays within each game are considered, with about 70 defensive plays per game. Further, the model is intended to be used throughout the season, so it is important to note that the model will grow in strength as the season progresses and more data is acquired. This leads to a less-than-ideal amount of samples, ranging from 70 to 700 depending on the point in the season. The number of features is the next hurdle, which stems from the difficulty of both live and preceding annotation. Some features like current yard line, down, and distance are immediately apparent after a play has concluded. However, other more powerful features are not available to be read until right before the snap as they develop very late, which does not give the user time to input them. Some examples of this include offensive formation, and motion (the presnap movement of offensive players). These features are shown to be more powerful in classic preparation by teams, with some formation and motion combinations indicating specific plays.

3.2.2 Time Constraints

Building on what was mentioned in the "Data Limitations" section, the features carried throughout the model's prediction must be entered and processed in an extremely short time. The play clock of a Canadian university football game allows for 20 seconds between the spot of the ball and the snap of the following play. This means that the user of the model that provides prediction has to enter in the data for the current play, then relay the prediction, then have the coach make the play call, then have the defense comfortably line up, all within that same 20 seconds. Therefore, features that do not require any additional analytics or thinking are required; ie down, distance, and yard line. Other previously mentioned more powerful features are unable to be entered, as the time between entry and the next play is so small.

3.3 Implications

Given the predictive power made available to coaches by this technique, both play calling and individual coaching will be improved. More play calls aimed at stopping the predicted plays will be called at the right time, which correlates to an increased chance of winning per game and in turn a better win-loss record. This allows teams with less talent in either coaching or players to perform better against a statistically better opponent, increasing parity and particularly lopsided conferences.

Chapter 4

Dataset

The dataset used for this study consisted of 10 games from the 2023 AUS Football season, with the same team being studied in depth and other teams used for supplementary analysis. The data of the team of focus is moderately imbalanced, with pass plays making up the majority as shown in Figure 4.1, and consisting of 820 samples altogether. Further, there are no clear and obvious relationships between the annotated features and play type as shown in Figure 4.2, apart from pass plays occurring more at further

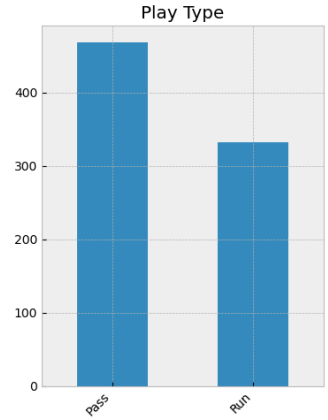


Figure 4.1: Class distribution of dataset

distances, and some yard line clustering. The defining patterns of play type for this team are clearly more complex. This drives the motivation to engineer additional features, and then to apply simple and complex predictive models to the data. The key distinction between features is annotated versus engineered — where annotated features must be easily and quickly observable in-game, engineered features are functions of the annotated features.

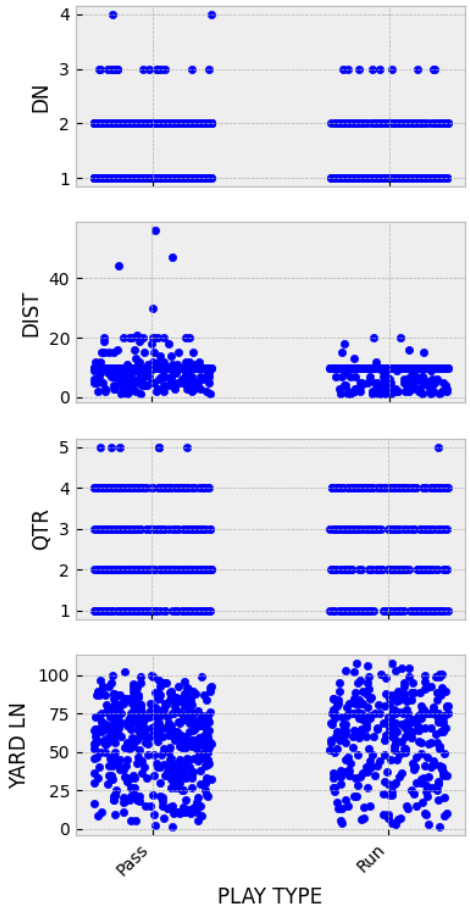


Figure 4.2: Basic relationships between class and annotated features in dataset.

4.1 Annotated Features

Annotated features are features that are inputted manually prior to a live scenario, and are simply data entry. These features would typically be inputted on a play-by-play basis using the previously mentioned tools like HUDL or DVSport, and then that third-party software would export the data to a spreadsheet. However, it is very important that annotated features are able to be inputted live, as the engineered features have no basis for creation without the live entry. Therefore, only features that can be feasibly entered within the short time frame between the spot and snap of the ball were considered.

4.1.1 Quarter

Denotes the quarter in game the current play has taken place in. Canadian university football is played in four 12-minute quarters. Therefore is of type *integer*, an ordinal variable, and in {1, 2, 3, 4}.

4.1.2 Down

Denotes the down of the current play. A 'down' is defined as an attempt by a team's offence to advance the ball toward the opponent's end zone. In Canadian football, the offence gets three tries, or 'downs', to advance the ball 10 yards from the spot of the first down marker, or from the line of scrimmage on subsequent downs. If the offence cannot advance the ball to the line of scrimmage within the three given downs, they turn the ball over to the other team. Shown in Figure 4.3 is the process of the offence attempting to advance the ball on 1st down and 10 from the 40-yard line, achieving a gain of 5 yards, then attempting to advance the ball on 2nd down and 5 yards. The

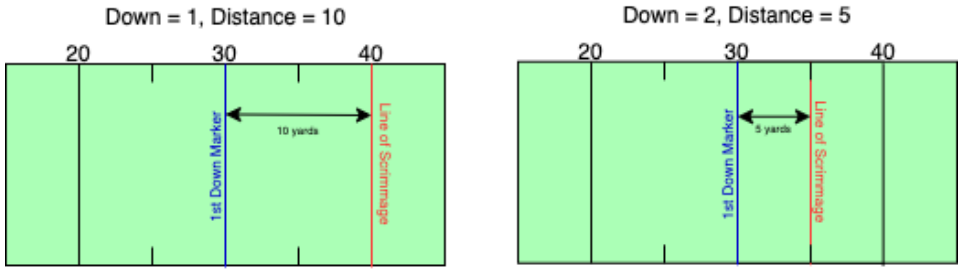


Figure 4.3: Diagram of the concepts of downs in football

feature 'down' is of type *integer*, an ordinal variable, and in {1, 2, 3}.

4.1.3 Distance

Distance is defined as the distance between the line of scrimmage and the first down marker; essentially it is the required distance for the offence to advance the ball to be awarded a new set of downs. This is shown in Figure 4.3, where in the left diagram the offence must advance the ball 10 yards, and in the second the offence must advance the ball 5 yards. The type of distance is *integer*, is a discrete numerical variable, and is in {0, 1, ... , 110}. The range has a maximum of 110, but distance is very typically between 1 and 20 yards.

4.1.4 Series

A series is defined as a sequence of consecutive plays within a game of football, synonymous with "drive". For example, consider an offense gaining possession of the ball at the negative 35-yard line. They are able to advance the ball all the way to the opposing (defensive) end zone, scoring a touchdown. Directly after this, the offence leaves the field to bring the extra point team on. The plays between the first play with the ball in their possession to the touchdown scoring play would be considered within the same

series or drive, so all those plays would be annotated with the same drive number. Another example would be the offense gaining position at the negative 10-yard line, failing to reach the 1st down marker on their 1st and 2nd down attempts, and bringing their punt team on for 3rd down. This series would then only contain *two* plays. If the play occurred on the first set of continuous plays, it would have $series = 1$. If it occurred on the seventh set of continuous plays, it would have $series = 7$. Series then is of type *integer*, a discrete numerical variable, and in {1, 2,..., n}, where n is the number of offensive series that a team had during a given game.

4.1.5 Points For

Points For is a simple count of points the offense has scored in the game before the current play. The scoring structure of Canadian football is as follows:

- Touchdown: 6 points
- Extra Point: 1 point
- Field Goal: 3 points
- 2-Point Convert: 2 points
- Safety: 2 points
- Rouge: 1 point

For example, if the offense has so far scored two touchdowns, two extra points, and one field goal, $points\ for = 17$. Points For is an *integer*, discrete numerical variable, and in $\{0, 1, ..., \infty\}; a team could theoretically score infinite points but almost all samples will have an upper bound$

4.1.6 Points Against

Points Against is a simple count of points the other team's offense has scored in the game before the current play. The scoring structure is defined in Section 4.1.5. Points Against is an *integer*, discrete numerical variable, and in $\{0, 1, ..., \infty\}; anopposingteamcouldtheoreticallyscorei$

4.2 Engineered Features

Engineered features are features that aren't entered directly in the live setting, but instead are computed as a function of the aforementioned live annotated features. Therefore, they do not have to be something easily observable, but are under the constraint that they only reference features from the annotated features listed in Section 4.1.

4.2.1 Yard Line

Yard line denotes where on the field the line of scrimmage (a function defined in Figure 4.3) is placed on the current play relative to the offence's and defence's end zones. In football the yard line is not a simple continuous range, as it is split at the middle of the field (55-yard line) and counted relative to the current offensive direction. As the offence advances past midfield, the yard line starts at 55 and decreases to 0 when the end zone is reached. Conversely before midfield, the yard line is marked as 0 from the offense's own end zone and decreases to negative 54 before reaching 55 at midfield. For example if the offense is 35 yards away from the opposing team's end zone, they are at the 35 yard line. If the offense is 75 yards away from the end zone (past midfield), they are on the negative 35 yard line. This is because they are 20 yards past the 55-yard line, so the yard line is negative and has a difference of 20 from the 55-yard line.

To combat this strange numbering scheme, the yard line feature was modified to reflect simply the distance from the opposing (defensive) team's end zone, still relative to the direction of the offense's advancing of the ball. The piecewise function of this transformation is as follows:

$$YL(x) = \begin{cases} x + 100 & \text{if } x < 0; \\ x & \text{otherwise} \end{cases}$$

In Figure 4.4 we see an example where the offense is currently on the negative 35-yard line, advancing left toward the defensive end zone. Before the transformation mentioned above, the yard line would be -35; this is shown in the upper diagram. After applying the transformation in the lower diagram, we can see that this has been converted to simply 75 yards away from the defensive end zone. Yard line is of type *integer*, a discrete numerical value, and in {0, 1, ..., 110}. [1]

4.2.2 Play of Series

Play of Series tracks the current play's place in a given series. For example, if an offense gained possession of the ball, ran one play on 1st down and now it's 2nd down, the current play would have $play\ of\ series = 2$. This is because it's the second play of the current drive. Mathematically, the feature is defined as the following:

$$f(i) = \sum_{j=1}^{i} g(i,j), \text{ where } g(i,j) = \begin{cases} 1 & \text{if } D(i) = D(j); \\ 0 & \text{otherwise} \end{cases}$$

- $f(x)$ is the function that computes play of series;

[1] Note: the yard line itself is annotated live as it is easily and quickly observable, but the transformation occurs separate from the data entry; thus, it is an engineered feature.

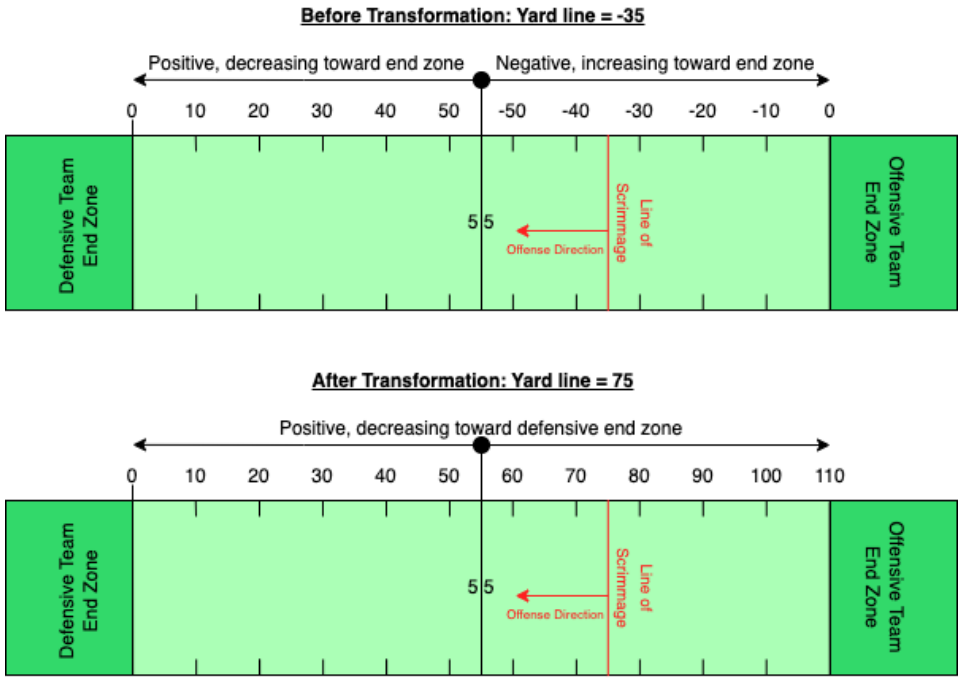


Figure 4.4: Yard line definition before and after transformation.

- $g(i, j)$ is the indicator function for each drive up to play i;
- D is the drive number.

4.2.3 Gain/Loss

Gain/Loss is defined as the difference between the current play's yard line and the previous play's yard line. The signifies the gain or loss of yards on any given play in the dataset. It is defined as:

$$f(i) = \begin{cases} 0 & \text{if } i = 0; \\ YDLN(i-1) - YDLN(i) & \text{otherwise} \end{cases}$$

4.2.4 Average Yards Per Run

Average yards per run tracks how many yards are gained/lost on any given run play during the game. For example, if the offense has tried three run plays and gained 7, 3, and -2, they have gained an average of 2.667 yards per run play. This feature is cumulative, meaning that as the game goes on, the current play considers only the previous plays from that game. So, on the first play, average yards per run would be zero. The fifth play would consider any run plays between the first and fourth play, and so on. Mathematically, this feature is defined as the following:

$$f(i) = 0 + \frac{\sum_{j=2}^{i-1} g(j)}{\sum_{j=1}^{i-1} h(i)};$$

$$g(i) = \begin{cases} YL(i-1) - YL(i) & \text{if } PT(i-1) = RUN; \\ 0 & \text{otherwise; and} \end{cases}$$

$$h(i) = \begin{cases} 1 & \text{if } PT(i) = RUN; \\ 0 & \text{otherwise.} \end{cases}$$

- $f(i)$ is the function that computes the average yards per run of the current play x. It takes the average as a fraction of two summations; which make up the average yards per run play up until the current play.

- $g(i)$ is the function that computes the gain or loss of the previous play if it was a run; otherwise, it returns 0.

- $h(i)$ returns 1 if the play i was a run, and 0 otherwise.

- YL is the yard line value for play i.

- PT is the play type value for play i.

Average yards per run is of type *integer*, is a continuous numerical variable, and in {-120, 120}.

4.2.5 Average Yards Per Pass

Average yards per pass measures the same as average yards per run, but for passing plays. For example, if the offense has tried three passing plays and gained 10, 5, and -5, they have gained an average of 3.333 yards per pass play. This feature is cumulative, meaning that as the game goes on, the current play considers only the previous plays from that game. Thus, on the first play, average yards per pass would be zero. The fifth play would consider any pass plays between the first and fourth play, and so on. This feature is defined as the following

$$f(i) = 0 + \frac{\sum_{j=2}^{i-1} g(j)}{\sum_{j=1}^{i-1} h(i)};$$

$$g(i) = \begin{cases} YL(i-1) - YL(i) & \text{if } PT(i-1) = PASS; \\ 0 & \text{otherwise; and} \end{cases}$$

$$h(i) = \begin{cases} 1 & \text{if } PT(i) = PASS; \\ 0 & \text{otherwise.} \end{cases}$$

- $f(i)$ is the function that computes the average yards per run of the current play x.

It takes the average as a fraction of two summations, which make up the average yards per run play up until the current play.

- $g(i)$ is the function that computes the gain or loss of the previous play if it was a run; otherwise, it returns 0.
- $h(i)$ returns 1 if the play i was a run, and 0 otherwise.
- YL is the yard line value for play i.
- PT is the play type value for play i.

Average yards per pass is of type *integer*, is a continuous numerical variable, and in {-120, 120}.

4.2.6 Score Differential

Score differential is a simple feature that reflects the differential between points for and points against. For example, if points for is 14 and points against is 7, score differential would be 7, as the offence would have a 7-point lead. If the offence was trailing by 7, the score differential would be -7. Score differential is defined as

$$f(x) = PF(x) - PA(x)$$

Score differential is of type *integer*, is a discrete numerical variable, and in $\{-\infty, \infty\}$.

4.2.7 Plays Since Points For, Points Against, Point Difference

This feature tracks how many plays it has been since the specified column (points for, points against, point difference) has changed. For example, if the opposing team just

scored and it's the third play of the offensive series directly following that score, plays since points against on this play would be *3*. Mathematically, this is defined as:

$$f(i) = \begin{cases} 0 & \text{if } i = 0; \\ g(i) & \text{otherwise; and} \end{cases}$$

$$g(i) = \begin{cases} 1 & \text{if } COL(i) \neq COL(i-1); \\ f(i-1)+1 & \text{otherwise.} \end{cases}$$

- $f(i)$ is the base case wrapper function, as we cannot reference the $i-1^{th}$ row when $i = 1$.
- $g(i)$ is the function that actually tracks changes in COL, where the count restarts at 1 once the value in COL changes.
- COL is the column referenced in the "Since" feature; for example, Points For.

4.2.8 Previous Play Result

Previous play result tracks the outcome of the previous play as a categorical variable, with the intuition that coaches call plays from a reactionary frame of mind. The categories are:

- $First\ Play\ of\ Series$: No previous data to display; thus, this is a category on its own.
- $Rushing\ 1st\ Down$: The offence attempted a run play, and was awarded a new set of downs by advancing the ball past the first down marker.

- *Passing* 1*st Down* : The offence attempted a passing play, and was awarded a new set of downs by advancing the ball past the first down marker.

- *Complete* : The offence attempted a passing play and the ball was caught, but short of the first down marker; thus, the gain was greater than zero but less than the distance to a first down.

- *Incomplete* : The offence attempted a passing play but the ball was not caught, resulting in a gain/loss of zero, as the ball will be spotted on the next down at the same yard line.

- *Sack* : The offence attempted a passing play, but the quarterback was tackled for a loss of yards. The gain/loss is less than zero in this case.

- *Rush* : The offence attempted a running play and the yards gained were short of the first down marker.

- TFL : TFL, or tackle for loss, implies that the offence attempted a rush play but was tackled for a loss of yards.

Mathematically this is defined as the following:

$$f(i) = \begin{cases} First\ Play\ of\ Series & \text{if } POS(i) = 1; \\ g(i) & \text{otherwise;} \end{cases}$$

$$g(i) = \begin{cases} p(i) & \text{if } PT(i-1) = PASS; \\ r(i) & \text{otherwise;} \end{cases}$$

$$p(i) = \begin{cases} Passing\ 1st\ Down & \text{if } GNLS(i-1) \geq DIST(i-1); \\ Complete & \text{if } GNLS(i-1) < DIST(i-1) \wedge GNLS(i-1) > 0; \\ Incomplete & \text{if } GNLS(i-1) = 0; \\ Sack & \text{otherwise; and} \end{cases}$$

$$r(i) = \begin{cases} Rushing\ 1st\ Down & \text{if } GNLS(i-1) \geq DIST(i-1); \\ Rush & \text{if } GNLS(i-1) < DIST(i-1) \wedge GNLS(i-1) \geq 0; \\ Sack & \text{otherwise.} \end{cases}$$

Chapter 5

Methodology

5.1 Process Summary

The overall process for this study has three main parts. Data preparation, where we engineer features, format and standardize the data, and perform oversampling. Model preparation is concerned with the hyperparameter optimization process, as well as general training. Model prediction is centered around the testing of the optimized models and analysis for results.

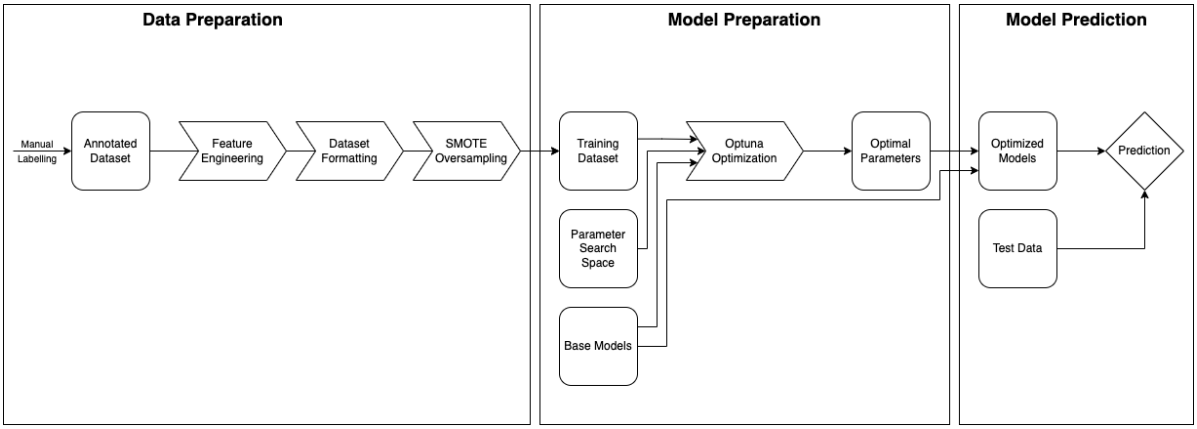


Figure 5.1: Overall Process Diagram

5.1.1 Data Preparation

Feature Annotation

Features listed in "Annotated Features" (Section 4.1) are manually inputted on a play-by-play basis by a team's associate to provide the basis for the dataset. This can be done either in third-party software like Hudl or DVSport and exported into a spreadsheet, or directly in spreadsheet software like Microsoft Excel, as shown in Figure 5.2.

Feature Engineering

Features listed in the "Engineered Features" are generated using the annotated data and created by using the functions defined in Section 4.2 for each engineered feature. The state of the data after adding engineered features compared to Figure 5.2 is shown in Figure 5.3.

Class Oversampling

To account for class imbalances between "Run" and "Pass" plays, we employed SMOTE to generate synthetic samples of the minority class. In this case, the minority class is

PLAY #	ODK	SERIES	QTR	YARD LN	HASH	DN	DOWN	DIST	PLAY TYPE	PF	PA
2	D	1	1	-28		1		10	Pass	0	0
3	D	1	1	-26		2		12	Run	0	0
8	D	2	1	-38		1		10	Pass	0	0
9	D	2	1	-43		2		5	Pass	0	0
14	D	3	1	-35		1		10	Pass	2	0
15	D	3	1	-35		2		10	Pass	2	0
22	D	4	1	-11		1		10	Run	2	0
23	D	4	1	-17		2		4	Run	2	0
24	D	4	1	-26		1		10	Run	2	0
25	D	4	1	-25		2		11	Pass	2	0
26	D	4	1	-52		1		10	Pass	2	0
27	D	4	1	-52		2		10	Run	2	0
29	D	5	1	30		1		10	Run	2	0
30	D	5	1	22		2		2	Run	2	0
31	D	5	1	15		1		10	Pass	2	0

Figure 5.2: Example Feature Annotation in Excel

SERIES	QTR	YARD LN	DN	DOWN	DIST	PLAY TYPE	GN/LS	PF	PA	PLAYOFSERIES	SCOREDIFF	SINCEPF	SINCEPA	SINCEPD	AVGRUN	AVGPASS	PREV PLAY TYPE	PREV GN/LS	PREV DIST	PREV RESULT
1	1	84	2		12	0	8	0	0	2	0	2	2	2	0	-2	1	-2	10	8
2	1	72	1		10	1	5	0	0	1	0	3	3	3	8	-2	0	8	12	6
2	1	67	2		5	1	0	0	0	2	0	4	4	4	8	1.5	1	5	10	3
3	1	75	1		10	1	0	2	0	1	2	1	5	1	8	1	1	0	5	6
3	1	75	2		10	1	7	2	0	2	2	2	6	2	8	0.75	1	0	10	5
4	1	99	1		10	0	6	2	0	1	2	3	7	3	8	2	1	7	10	6
4	1	93	2		4	0	9	2	0	2	2	4	8	4	7	2	0	6	10	4
4	1	84	1		10	0	-1	2	0	3	2	5	9	5	7.66666667	2	0	9	4	2
4	1	85	2		11	1	27	2	0	4	2	6	10	6	5.5	2	0	-1	10	7
4	1	58	1		10	1	0	2	0	5	2	7	11	7	5.5	6.16666667	1	27	11	1
4	1	58	2		10	0	4	2	0	6	2	8	12	8	5.5	5.28571429	1	0	10	5
5	1	30	1		10	0	8	2	0	1	2	9	13	9	5.2	5.28571429	0	4	10	6
5	1	22	2		2	0	7	2	0	2	2	10	14	10	5.66666667	5.28571429	0	8	10	4
5	1	15	1		10	1	0	2	0	3	2	11	15	11	5.85714286	5.28571429	0	7	2	2
5	1	15	2		10	1	0	2	0	4	2	12	16	12	5.85714286	4.625	1	0	10	5
6	2	74	1		10	1	8	5	0	1	5	1	17	1	5.85714286	4.11111111	1	0	10	6
6	2	66	2		2	0	7	5	0	2	5	2	18	2	5.85714286	4.5	1	8	10	3
6	2	59	1		10	0	0	5	0	3	5	3	19	3	6	4.5	0	7	2	2
6	2	59	2		10	1	10	5	0	4	5	4	20	4	5.33333333	4.5	0	0	10	4

Figure 5.3: Dataset after adding engineered features.

run plays. Synthetic Minority Over-sampling Technique (SMOTE) has been shown to provide substantial performance gains on heavily imbalanced datasets by Chawla [3], and its improvement on the semi-imbalanced dataset in this study will be monitored. At a high level, synthetic samples are generated by creating data points equidistant from multiple real minority class samples. This is illustrated visually in Figure 5.4, and leads to a balanced dataset. [1]

Data Formatting

Our first step in formatting the data is one hot encoding categorical features, or features that are non-numerical and have no ordinal properties. One hot encoding was shown by Seger [17] to outperform other categorical feature encodings in classification tasks, at the expense of increased dimentionality of the dataset. This problem is not prevalent in this study, as the number of features never reaches a cumbersome amount. The final step in formatting the data pertains to standardization. All features are transformed to have a mean of $\mu = 0$ and a unit variance of $\sigma = 1$. For each feature x_i, we define a transformed feature $z_i = \frac{(x_i - \mu)}{\sigma}$, ie x_i is an instance of feature x, μ is the mean of

[1] Note: SMOTE oversampling is only applied to lower data, as applying it to test data would inflate performance metrics.

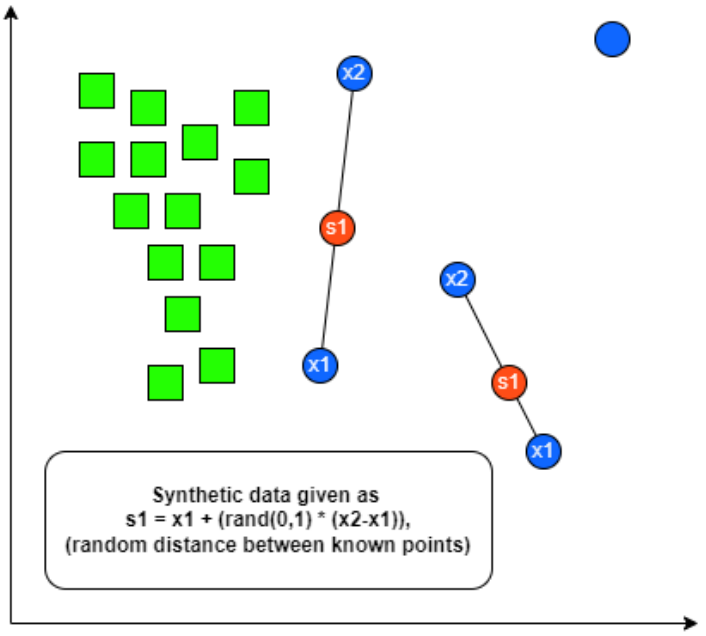


Figure 5.4: Example of SMOTE Oversampling

feature x, and σ is the variance of feature x. This is achieved with the SkLearn tool StandardScaler.

5.1.2 Model Preparation

Model Optimization

All models outlined in Section 5.2 are then optimized using the finalized training data, along with the parameter search space from Section 5.3.2 using the Optuna optimization method outlined in Section 5.3.1.

Model Training

After the optimized hyperparameters are obtained, models are created using those hyperparameters for use in the two types of analysis. Thus, different training data is used

for either the k-fold analysis or the application analysis.

5.1.3 Model Prediction

Two types of prediction occur in this study:

- K-Fold Analysis
  - The dataset is split into training and testing datasets consistent with k-fold analysis;
  - The test dataset is passed through the optimized and trained models, which output a single binary prediction: run or pass.
- Application Analysis
  - On each new play occurrence:
    * Annotated features are logged live;
    * Engineered features are generated automatically;
    * The sample is then formatted with one-hot encoding and standardization
    * The sample is passed into optimized and trained models outputting a binary prediction: run or pass.

5.2 Models

5.2.1 Dummy Classifier

The first model for this study is the dummy classifier implemented by the SciKitLearn Python library, which always predicts the majority class. The purpose of its inclusion is

to provide a baseline for our analysis, as it establishes a reference point for the performance of other models. This gives context to the performance of more complex models employed, and further both allows us to gain insight to the challenges posed by the class imbalance of this dataset and aids in a comprehensive analysis of the models used in this study.

5.2.2 Logistic Regression

Logistic regression is defined as 'statistical models which describe the relationship between a qualitative dependent variable (that is, one which can take only certain discrete values, such as the presence or absence of a disease) and an independent variable' [11]. The model's documented success in binary classification, interpretability with respect to feature importance, and computational efficiency are all reasons for its selection. Logistic regression is not suitable for high-dimensional data, but this does not concern our dataset.

5.2.3 Decision Tree

A simple decision tree is the third model used in this study. It classifies samples by recursively partitioning data until a leaf node is reached. Much like logistic regression, decision trees are able to handle both numerical and categorical data, but with the added benefits of basic explainability and visualization. An example of how this model may look is seen in Figure 5.5. In the case of this study, a decision tree was built using HUDL game reports to emulate a secondary baseline. This decision tree is a hypothetical implementation of what's available today, and provides further validation for the models selected.

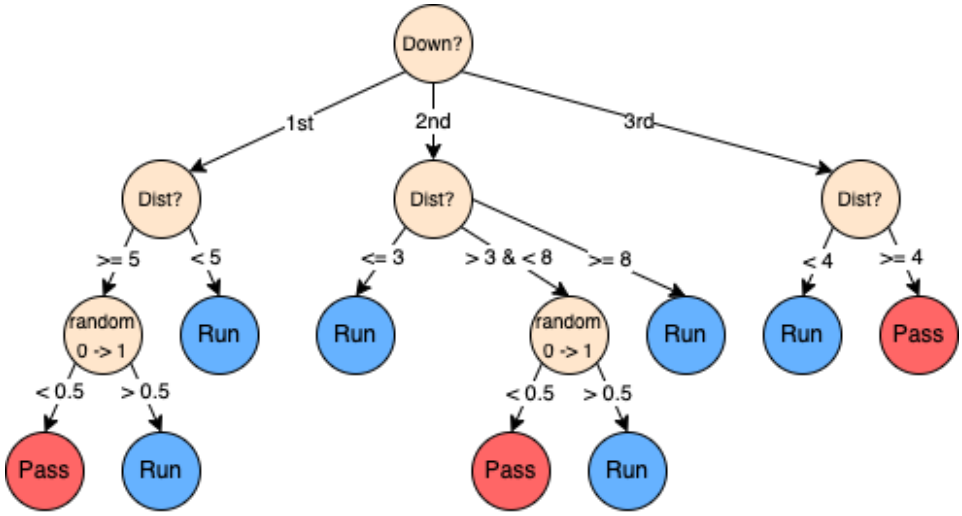


Figure 5.5: Architecture of tree derived from HUDL report data.

5.2.4 K Nearest Neighbours

K-nearest neighbours (KNN) classifies samples based on which training samples are closest to it within the feature space, with a majority vote from the k closest considered. Its inclusion was based on KNN's simple implementation and ability to capture spatial relationships within the data. A high-level depiction of how this model works is shown in Figure 5.6. Here the *blue* dot would get classified as *green*, as there are more *green* dots than *red* dots in the k nearest neighbours when $k = 5$. The drawback to KNN is its sensitivity to irrelevant features, thus a comprehensive feature selection process would benefit it greatly.

5.2.5 XGBoost

XGBoost is an ensemble machine learning method that iteratively builds decision trees that correct preceding iterations. Its benefits include the ability to model complex relationships in the data, in addition to advanced regularization techniques to prevent

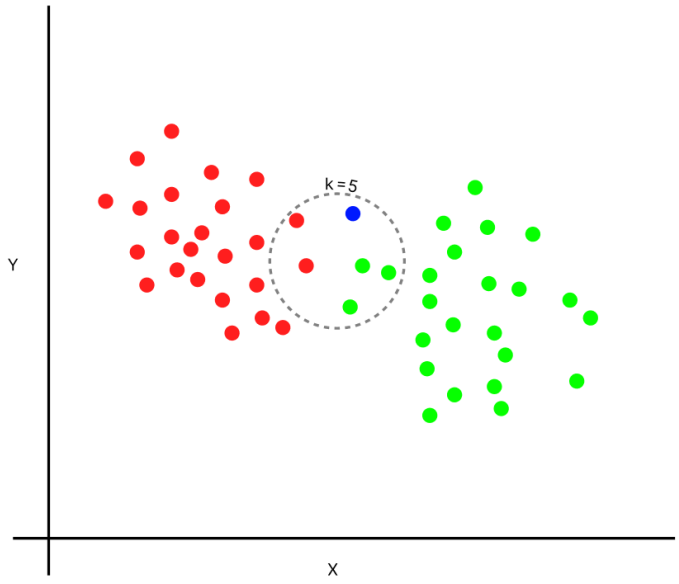


Figure 5.6: Simple visualization of KNN algorithm.

overfitting. These benefits led to its inclusion in this study. Its powerful classification methods have been explored in a variety of different domains, starting with its inception by Chen in 2016 [4].

5.2.6 Shallow Neural Network

Neural networks are known for capturing extremely complex patterns in data while remaining a black box for explainability. Shallow neural networks in particular are defined as neural networks with minimal layers overparameterized with a number of parameters greater than the number of samples in the dataset. Most models will struggle with such classification tasks, as there are many rules that are very non-basic and require increasingly complex parameters to identify. These shallow networks have been proven able to classify even random noise [13], and given this ability to fit data even without defined relationships, the belief is it will be able to detect all complex rules. The one

caveat that must not be overlooked is the tendency to overfit, and that will be monitored closely. A simple architecture of this type is shown in Figure 5.7, where there are n neurons in the input layer, m in the hidden layer, and $m > n$.

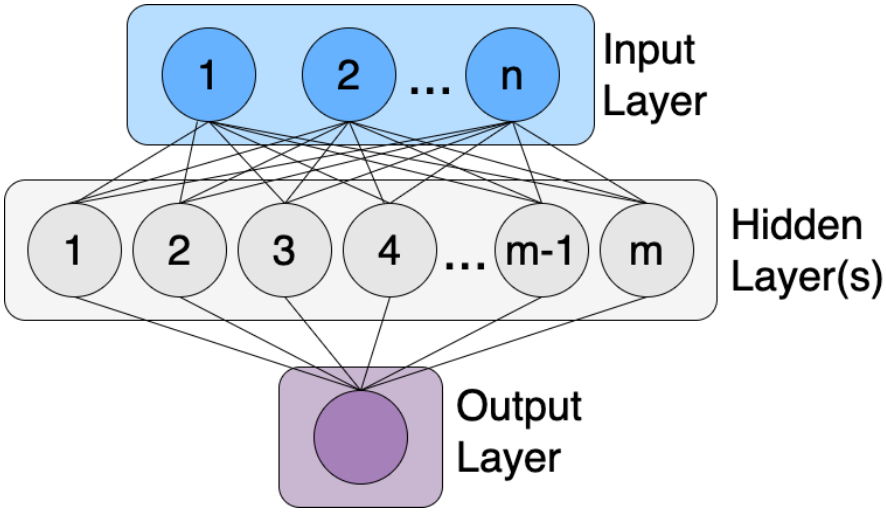


Figure 5.7: Simple shallow neural network architecture.

5.3 Model Optimization

All of the chosen models use a wide variety of hyperparameters, all of which must be tuned according to the data being passed in. This was illustrated by van Rijn [19], where specific hyperparameters were shown to have vastly different importance levels across datasets. The need for correct tuning of these models was explored further by Probst [14], where even marginal changes in hyperparameters yielded large performance gains. The unexplored domain of data used in this study lead to the choice of using a hyperparameter optimization engine; in our case, the Python optimization library Optuna. The overarching goal of this step is to find what configuration of the models

perform best under the testing conditions outlined in the Application Analysis and Theoretical Analysis of Section 5.4.

5.3.1 Optuna

Some of the models employed have upwards of ten parameters that can be tuned to aid performance, and so the search space for an optimal implementation can grow both quickly and significantly. More basic models like a grid search or random search have been commonplace in study [21], but are hindered by inefficiency and the inability to find the true optimal configuration. Optuna allows the search space to be dynamically constructed, where relational sampling is made possible. For example, with logistic regression, only some penalty types are applicable for specific solving algorithms: the Newton Cholsky solver can only use penalty l2, whereas the Saga solver can use l1, l2, and elasticnet. This is made possible by Optuna's define-by-run API, outlined in more detail by Akiba [1]. Further, it allows for pruning of less successful trials, allowing for more computational power to be spent on more promising trials. The specific strategy used is hyperband pruning as outlined by Li et al. [10], which organizes trials into specific bands based on the parameters used and performs successive halving on those bands of trials. Overall, the choice of Optuna allows for consistent implementation across models while efficiently finding the optimal configuration for the models used on this dataset.

5.3.2 Optimization Parameters

Each model was sent through an optimization study containing 1000 trials, where the objective function was average accuracy when testing the model using k-fold validation with $k = 5$ on the traning dataset containing both annotated and engineered features.

Early pruning occurred in trials as outlined above, and the final set of parameters was used in the actual analysis of each model [2].

Logistic Regression

- Solver: Refers to the algorithm used for optimizing the cost function. {*lbfgs, newton-cg, newton-cholesky, sag, saga*}.
- Penalty: The penalty term added to large coefficients, used to prevent overfitting. {*None, L1, L2, elasticnet*}.
- C: The inverse of regularization strength, where a smaller value keeps coefficients small to aid generalization. The trial suggests a float between {*0.5, 5*}.
- Max Iterations: The maximum iterations allowed for the solvers to converge. Integer between {*50, 5000*}.

K Nearest Neighbours

- N Neighbours: Number of neighbours considered in prediction. Integer between {*1, 50*}.
- Weights: The type of weights assigned to prediction; either uniform weights for each data point, or the inverse of distance from each data point. {*uniform, distance*}.
- Algorithm: Algorithm used to compute the nearest neighbours. {*ball tree, kd tree, brute*}.

[2]The full details of each parameter for each model will not be explained in full detail, as they do not fall directly within the purpose of this study.

- Leaf Size: Leaf size passed to ball tree or kd-tree algorithm, which affects speed of construction and memory requirements. Integer between {*5, 75*}.

- P: Power parameter for Minkowski distance, where $p = 1$ is Manhattan distance and $p = 2$ is Euclidean distance. Integer between {*1, 5*}.

XGBoost

- N Estimators: Number of boosting strategies to be performed. Integer between {*25, 1000*}.

- Loss: Loss function to be optimized. {*log loss, exponential*}.

- Criterion: Quality metric for splits. {*friedman MSE, squared error*}.

- Minimum Samples Split: Minimum samples required to perform a split on an internal node. Integer between {*2, 20*}.

- Learning Rate: The factor by which each tree's contribution is diminished. Float between {*0.001, 1*}.

- Max Depth: The maximum depth of any given regression estimator. Integer between {*2, 30*}.

Shallow Neural Network

- N Layers: The number of layers of neurons in the network. Integer in range {*1, 5*}.

- Layer Units: The number of units/neurons in each layer respectively. Integer in range {*32, 1024*}. [3]
- Dropout Rate: The rate of neurons set to zero on each epoch to ensure regularization. Float between {*0.05, 0.5*}.
- Optimizer: The optimizer responsible for updating the weights of neurons during the training process. {*Adam, RMSProp, Stochastic Gradient Descent*}.
- LR: Determines step size taken in optimization function. Float between {*0.000001, 0.1*}.

5.4 Analysis of Performance

Analysis of the models will be split into two sections, with the "Theoretical Analysis" taking a more classical approach and "Application Analysis" aimed at the real-world performance of the proposed model.

5.4.1 Theoretical Analysis

K-fold cross-validation is a widely employed technique in machine learning for robust model evaluation, with proven success in a wide variety of domains from mineral classification [2] to software fault detection [15]. Instead of relying on a single train-test split, this method partitions the dataset into emphk subsets or "folds", as seen in Figure 5.8a. The model is trained and validated emphk times, each time using a different fold as the test set and the remaining folds as the training set. This process helps mitigate

[3]Note: each layer has its own unit parameter, making this a variable number of parameters.

the impact of variability in a single random split, providing a more reliable estimate of the model's performance. The final evaluation of accuracy is obtained by averaging the results from the k iterations, yielding a more representative measure of the model's generalization capabilities. Formally, the definition of accuracy used is the following:

$$A(K) = \frac{1}{K} \sum_{i=1}^{K} A_i$$

- $A(K)$ is the overall accuracy of k-fold validation with k splits;
- K is the number of splits; and
- A_i is the accuracy with the i^{th} fold as the test set and the other $K - 1$ splits as the training set.

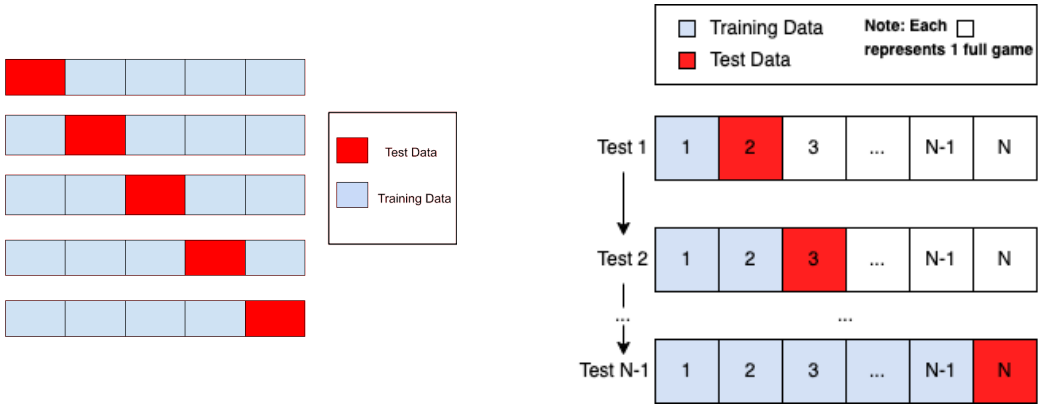
K-fold cross-validation is especially beneficial when dealing with limited data, contributing to a more comprehensive understanding of the model's effectiveness across diverse subsets of the dataset. This is very important in the case of our dataset, where we have very limited samples and features based on the aforementioned data limitations.

5.4.2 Application Analysis

In the case of application, we are assuming the model will be used throughout an entire season. Thus, our evaluations are made with each game being a specific point in testing. The first test would be with the first game as the training data and second game as the test data. The next would have the first and second games as the training data, and third game as the test set, and so on. This assumes that, in practice, the model would have access to all preceding games that season as training data and the current game it's being used on would then be the test data, as shown in Figure 5.8b. After

the performance from each of the tests is calculated, both the mean accuracy along with trends observed over the course of the season will be reported. Further, the model is at its most useful in scenarios where the type of play is not immediately clear. For example, any given 1st down and 10, or a 2nd down and medium distance (4-8). This is contrasted with more obvious scenarios, like 2nd down and 10+. In these scenarios, it is common knowledge in football that the offence will choose to pass the ball. Pass plays are generally an all-or-nothing play, where run plays offer a smaller amount of yards but are far more consistent. Another example would be 3rd down and 1, where offences are extremely likely to run the ball. These factors will allow us to make assumptions about the models performance in less obvious scenarios. Accuracy metrics will be further split into a table with the following columns:

- Down $\in \{1, 2, 3\}$.
- Distance $\in \{1-2, 2-7, 7+\}$, to represent short, medium, and long distances.
- Quarter $\in \{1, 2, 3, 4\}$.



(a) K-fold Validation Method

(b) Application Testing Method

Figure 5.8: Comparison of the two evaluation methods used.

Chapter 6

Results and Discussion

6.1 Model Optimization

The optimization process showed varying convergences, with XGBoost reaching its most optimal much later in the process and logistic regression reaching its optimal parameters early on. This is shown in Figure 6.1. After completing 500 trials, the following parameters were the most successful:

- Logistic Regression
  - Solver: Newton CG
  - C: 1.799
  - Max iterations: 2268
- KNN
  - Neighbours: 47
  - Weights: Uniform

  - Algorithm: KD Tree
  - Leaf Size: 60
  - P: 2.542

- XGBoost
  - Estimators: 213
  - Loss: Log Loss
  - Criterion: Friedman MSE
  - Min Samples Split: 2
  - Learning Rate: 4.319

Figure 6.1: Optimization accuracy over the 500 trials, by model.

  - Max Depth: 3

- Neural Network
  - Optimizer: Adam
  - Learning Rate: 0.0001
  - Number of layers: 4
  - Layer Neurons: {144, 187, 198, 621}
  - Layer Dropout: {0.19, 0.45, 0.48, 0.10}

6.2 K-Fold Results

Interestingly, the inclusion of engineered features aided the performance of some models and hindered the performance of others. Notably, in Table 6.1, logistic regression and XGBoost saw significant improvement with, KNN and the neural network actually performing worse. Furthermore, the only models that outperformed the dummy majority classifier were the logistic regression and XGBoost models, the only ones that benefited from the feature engineering. The baseline created from the HUDL reports performed second-worst, only outperforming the KNN model. When we dived into the results by down and distance, which is the typical breakdown in football tendency, a few things become apparent. In Figure 6.5b, we observe that despite having identical accuracies, XGBoost and logistic regression achieved their performance in slightly differing ways. In more "trivial" scenarios like 2nd and long distance and 3rd and short distance, a team is expected to pass and run respectively. The logistic regression outperformed XGBoost in these scenarios, being 16% more effective on 3rd and short and 1% better on 2nd

and long. On less obvious scenarios, XGBoost performed best, scoring 5% better in 2nd and medium scenarios and 1% better on 1st and long. Other observations include the neural network's performance on 3rd down in general, which outperforms all models on medium and long distance.

	pre-eng accuracy	post-eng accuracy	accuracy delta
Dummy	0.563636	0.563636	0.000000
LR	0.615152	0.636364	0.021212
KNN	0.553030	0.551515	-0.001515
XGBoost	0.559091	0.639394	0.080303
PyTorch	0.486364	0.490909	0.004545
Napkin Tree	0.565152	0.556061	-0.009091

Table 6.1: Classification accuracy by model, before and after feature engineering

6.3 Application Results

In the application testing, there was far more variance in the top-performing model throughout a given season. As observed in Figure 6.3, in about a third of all trials, the majority baseline was the most successful. Early on, the decision tree from HUDL reports was fairly successful, achieving the best performance in three straight trials. However as

the data from the season grew, each of logistic regression, XGBoost, and KNN began to improve. Overall, models benefited from the engineered features throughout the trials, with a sharp increase in improvement from the seventh trial onward, shown in Figure 6.4. This shows the potential for drastic improvement as the season continues, especially for teams that play more games in a season. XGBoost had the best performance by the end of the season, with 79.7% accuracy on the final trial shown in Table 6.2. Further, no models showed a negative impact from the introduction of the engineered features, and XGBoost as the most performant model actually benefited the most, at 34.8%. With regards to down and distance, the same trend from k-fold testing arises. Logistic regression performs the best in obvious scenarios, like 2nd and long and 3rd and short, whereas XGBoost performs better in less obvious ones like 2nd and medium and 1st and long. This observation, along with the ones made in the k-fold analysis section, solidify that the engineered features make the biggest difference for identifying more nuanced tendencies in less obvious scenarios, but is less required for obvious ones. It follows then that additional thought-out engineered features would further benefit from predictions of this nature. However, it is worth mentioning that some special situations that arise in football could be the root cause of confusion; for example, after a team has secured a win on defence, they may elect to "kneel out" the game. This is defined as deliberately taking a knee to take time off the clock and avoid turning the ball over. This would skew predictions, as regardless of situation, the team will run a running play for a loss of 1.

	pre-eng accuracy	post-eng accuracy	accuracy delta
Dummy	0.753623	0.753623	0.000000
LR	0.594203	0.652174	0.057971
KNN	0.550725	0.739130	0.188406
XGBoost	0.449275	0.797101	0.347826
PyTorch	0.594203	0.623188	0.028986
Napkin Tree	0.579710	0.492754	-0.086957

Table 6.2: Accuracies by model on the last trial.

6.4 Comparisons to Existing Work

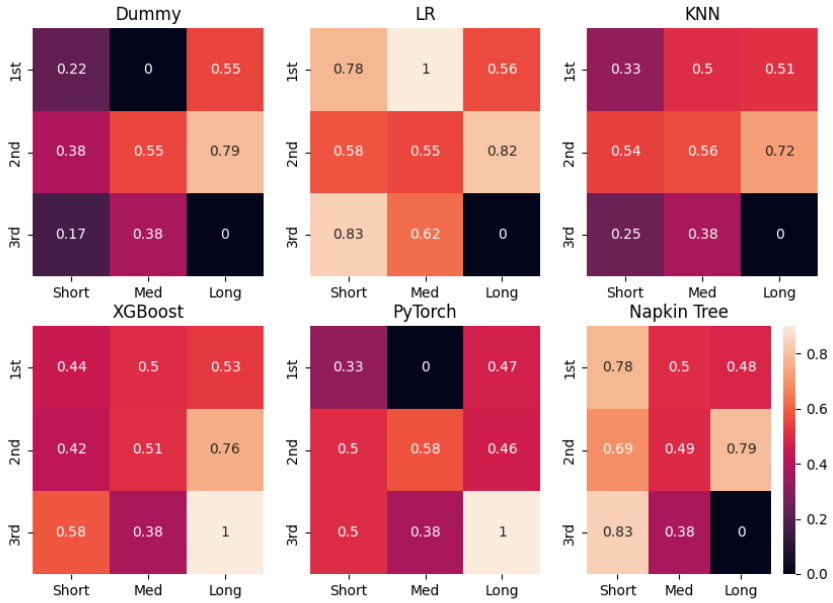
When contrasted by the study conducted by Joash [8], we see an improvement of about 5%. The top performing model of that study achieved 75.3% accuracy making use of 130,344 samples and more features than this study, most notably formation. [1] The top performing model of our study achieved 79.8% accuracy, with 820 samples and a more restricted set of features. Similarly, the models we used outperformed Markov models used by Otting [12], which had a best of 77.9%. This study further mentioned the variance in predictive power between teams, so a useful exercise would be testing the method outlined on data from other teams in university football.

6.5 Feature Importance

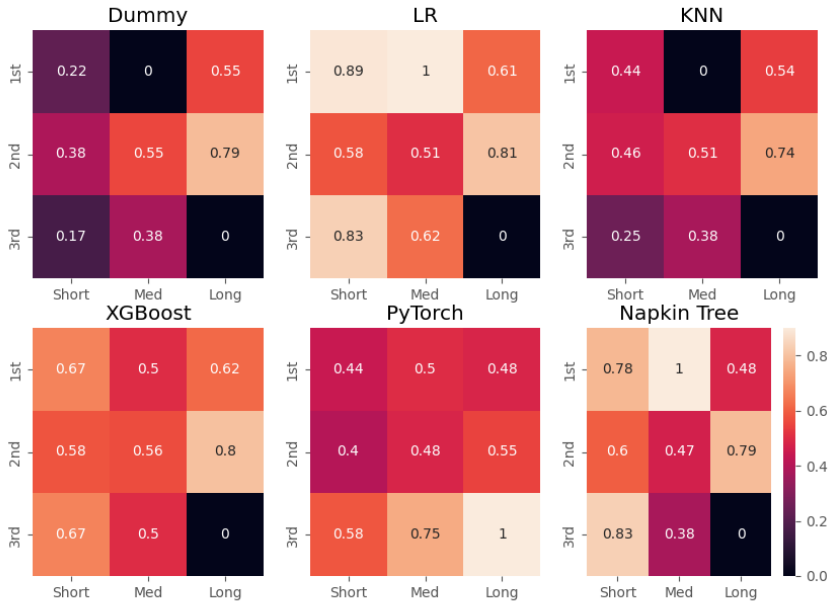
The decision of features differed in both choice and magnitude across models. What remained consistent, however, was that "DIST" was by far the most useful indicator. It is observed that in Figure 6.6, the logistic regression model made use of the "PREV RESULT" feature, but less so the other engineered features. This speaks to its ability to identify play type in obvious scenarios versus non-obvious ones. To contrast, XGBoost prunes insignificant features automatically painting a clear picture of what was useful to the model. In Figure 6.7, the model considers far fewer features than logistic regression. "PLAYOFSERIES" is the prevalent engineered feature differing from logistic regression's "PREV RESULT"; this also indicates the importance of these features in the context of ambiguity of the prediction scenario. Interestingly, XGBoost pruned all features

[1] Formation was omitted from this study as it only becomes clear right before the snap of the ball, making it entirely impossible to annotate in a live game setting.

related to “DN”, the current down of the play. Logistic regression had these features as some of the most important. With respect to features related to points, both models maintain that “PF” is far more important than ”PA”; with XGBoost omitting “PA” and “SINCEPA” altogether. Finally, “YARD LN” was largely unimportant to most models, showing that field position has less of an effect on tendency than previously thought. With respect to the neural network, we see similar trends in feature importances, with the exception of downs. Shown in Figure 6.8, downs 1 and 2 were instrumental in the neural network’s prediction, both positively and negatively for run and pass predictions. Distance typically swayed predictions toward passing plays, as when distances get longer to the first down, it’s more likely that a team will pass. Other notable observations include both “PF” and “PA” contributing positively to pass predictions, indicating pass plays are more depending on score when using this model. The importance of features is observed further in Figure 6.9, where we compare the decision boundaries of important versus unimportant features. It is made clear that the relationships between important features are far more interpretable than unimportant ones, with the model having much more success distinguishing samples in less complex decision boundaries.

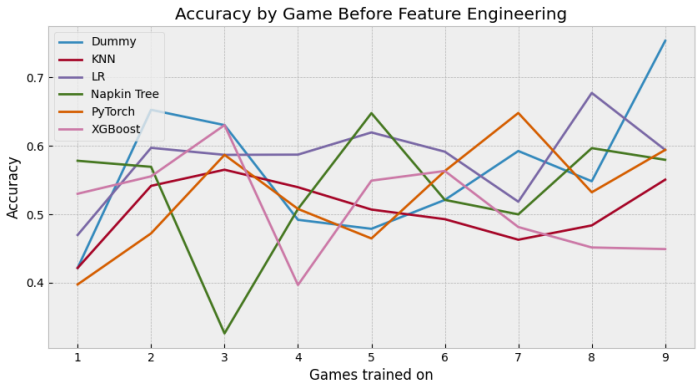


(a) Without engineered features.

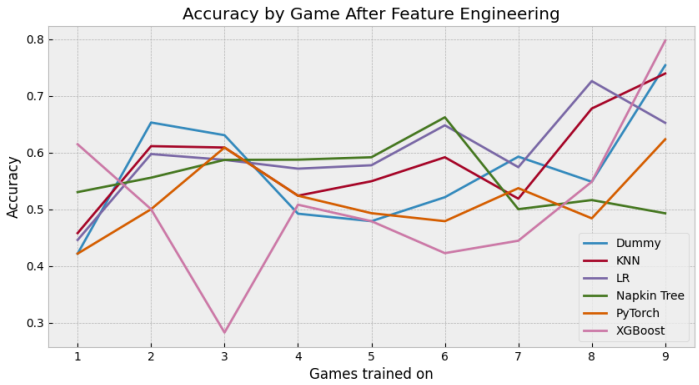


(b) With engineered features.

Figure 6.2: Accuracies by down and distance of each model.



(a) Accuracies without engineered features.



(b) Accuracies with engineered features.

Figure 6.3: Accuracies by game before and after feature engineering.

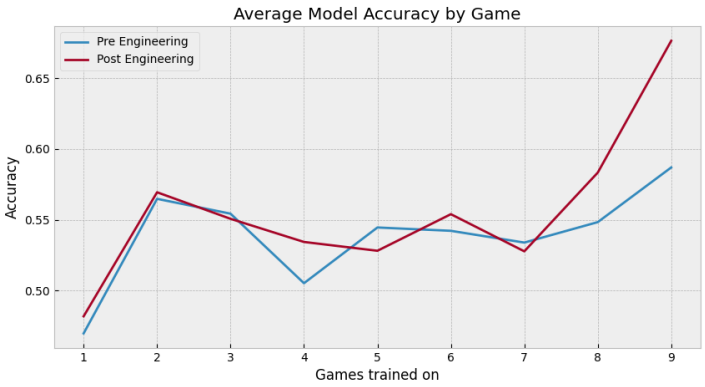
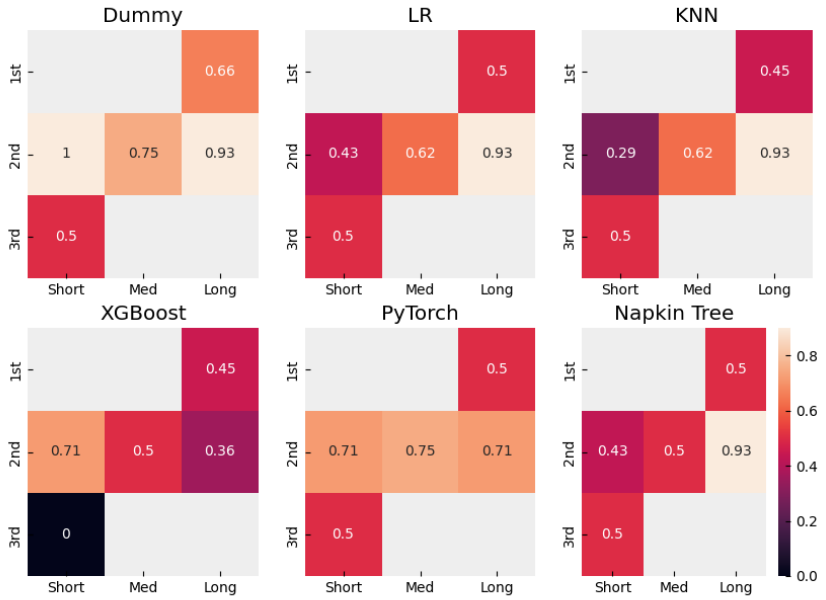
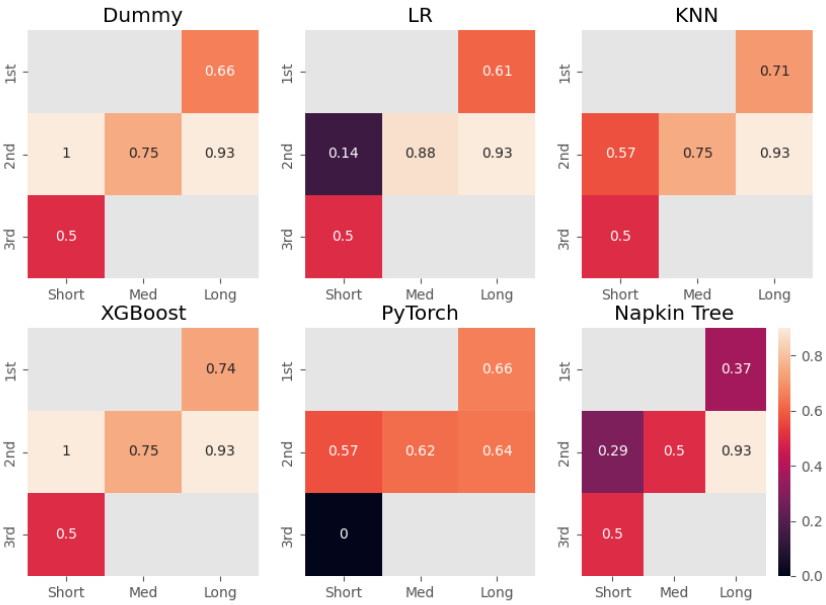


Figure 6.4: Accuracies before and after feature engineering.



(a) Without engineered features.



(b) With engineered features.

Figure 6.5: Accuracies by down and distance of each model on the final trial.

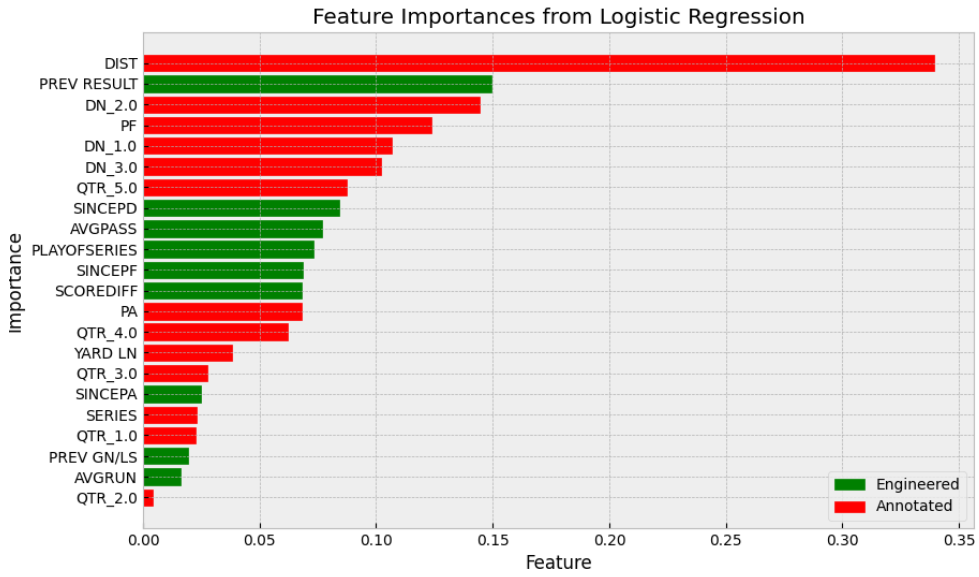


Figure 6.6: Importance by feature of logistic regression model.

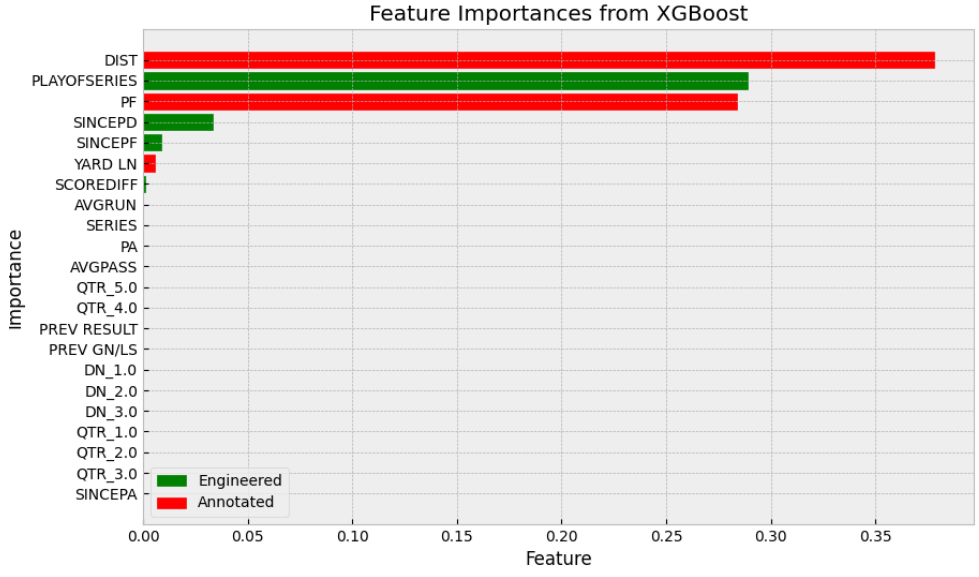


Figure 6.7: Importance by feature of XGBoost model.

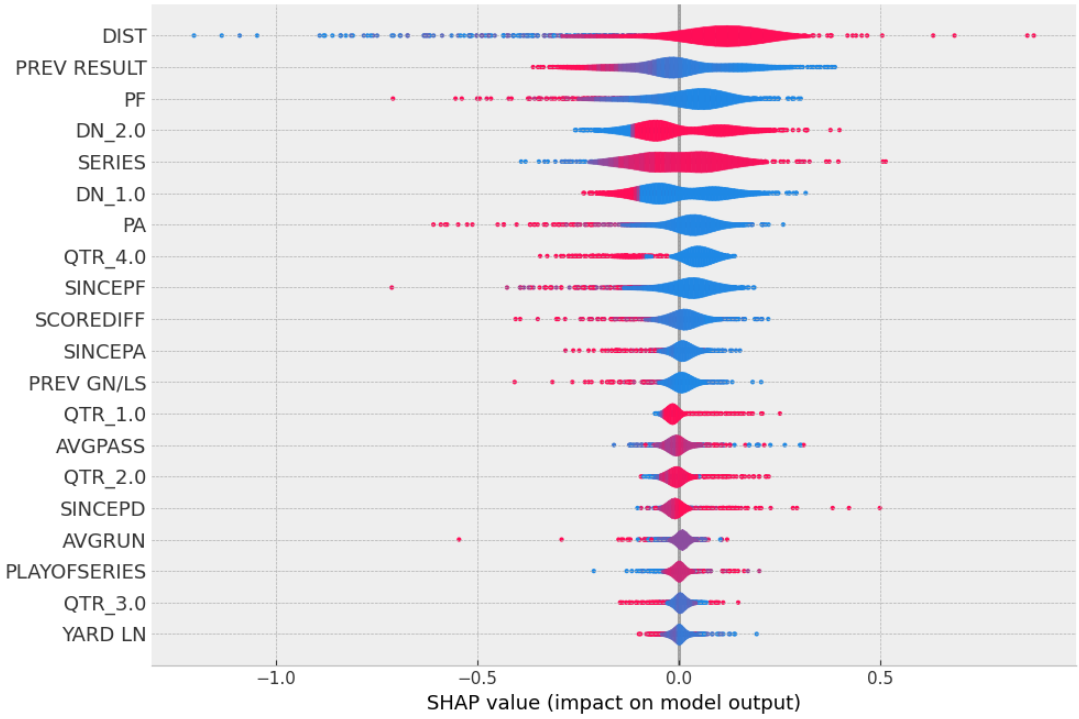
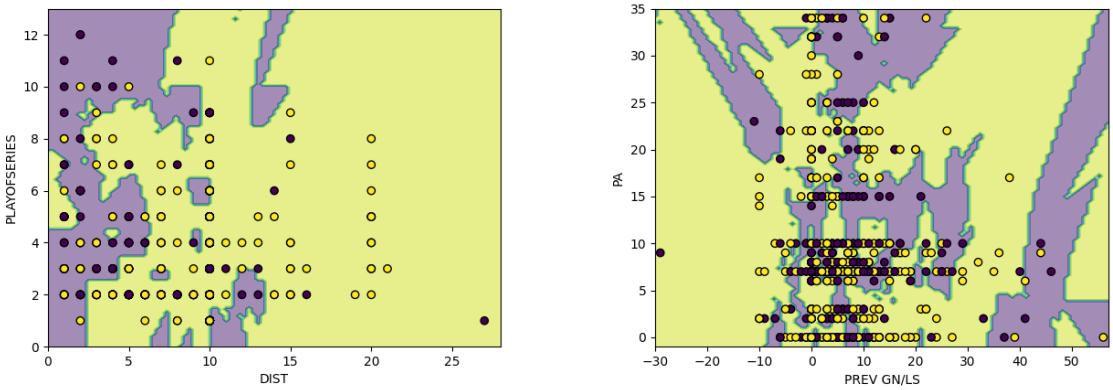


Figure 6.8: SHAP values of neural network; left indicates run prediction, right indicates pass prediction.



(a) Decision boundaries for KNN on "DIST" and "PLAYOFSERIES" features.

(b) Decision boundaries for KNN on "PA" and "PREV GN/LS" features.

Figure 6.9: Contrast of "important" vs. "unimportant" feature relationships in regards to decision boundaries.

Chapter 7

Conclusions and Future Work

7.1 Summary of Findings

In this study, we presented a novel method for play call prediction with limited game data. The models made use of functional engineered features as well as hyperparameter optimization. Despite extremely limiting data constraints, the best models outperformed the baseline methods, achieving an accuracy of 80% on the final game of the 2023 AUS football season. Further, ways to engineer features beyond simple methods were explored, and their impacts were apparent; for example, "PREV PLAY RESULT" which tracked the outcome of the previous play categorically, showed drastic importance in prediction. A difference in contextual accuracy was also observed, with XGBoost performing best in non-obvious scenarios and logistic regression performing best in obvious ones. This speaks to the difference in relationships given the current context. Important features were non-consistent across models, with the exception of "DIST" being the most influential feature for each model.

7.2 Limitations and Challenges

Despite moderate success, the data limitations still lingered throughout the study. Being limited in both samples and features had a strong negative effect on model performance. Further, because of the short time frame between annotation and the snap of the ball, the set of annotated features had to remain small and concise, further hindering performance. Unforseen circumstances in the dataset also skewed prediction, for example, with quarterback kneels and desperation plays. These behaviors fall into a different category of prediction, as they are trivial but the opposite of what would be expected; had these been accounted for, it follows that improved performance would come. The challenge of defining the search space for hyperparameter optimization was aided by following industry standard, though this could be sub-optimal based on the unorthodox data. The nuanced decision making of offensive coordinators caused most models to have difficulty identifying patterns that pertain to either class, especially in early trials. Further, other factors outside of the data are not accounted for; for example, injury, or the effect of weather patterns in the game. If a team that is very pass-heavy loses their starting quarterback, the models have no knowledge of this and will still predict pass with a high probability.

7.3 Implications

With the information and results from this study, we can give coaches the predictive power that gives their team's defence an edge over opposing offences. This improved defensive performance can potentially lead to more wins and greater overall success for a team, as it improves their win percentage. Further, as one of the first studies to

look at in-game data specifically, this study opens the door for others to research this topic further and provide improvements or modifications to the current process to aid performance.

7.4 Future Work

There is plenty of room for improvement for the current process outlined. Following work by Joash [8], adding madden ratings (ratings from 50-100 attributed to players for the NFL branded video game) to predictive models as well as derived models by position showed improvement. A similar skill metric for position groups at the university level could aid models further in this use case. Live-training in game is another potential improvement, where more recent games also carry a heavier weight in prediction. Another potential improvement is a stacked meta classifier, as described by Sesmaero [18]. In this case, the weights of the different model's predictions are based on their performance in specific situations; that is more weight is given to XGBoost when the down and distance is 2nd and 5. Ultimately, the largest improvement in this area of research will likely come from additional thought out features from a football context, as shown by the sharp increase from engineered features as the training set grew. With additional research being put into this domain, the power behind these models could create massive change in the university football landscape with regard to preparation and play calling.

7.5 Conclusion

Predicting the next play call in Canadian football is as much a human psychology task as it is a machine learning one; thus, the countless nuances in decision-making present a very difficult classification task. Despite this, 80% accuracy was achieved, proving the use of predictive models in live game settings is indeed viable. The results and developments from this study seek to guide and fuel deeper research into in-game prediction from a general team perspective as well as outside of football. With further analytics, the potential improvement given to teams is immeasurable.

Bibliography

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[2] Nurdan Akhan Baykan and Nihat Yılmaz. A mineral classification system with multiple artificial neural network using k-fold cross validation. *Mathematical and Computational Applications*, 16(1):22–30, 2011.

[3] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[4] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[5] Frank Cosentino. *Canadian Football: The Grey Cup Years*. Lulu. com, 2017.

[6] Michael Hamilton, Phuong Hoang, Lori Layne, Joseph Murray, David Padget, Corey Stafford, and Hien Tran. Applying machine learning techniques to baseball pitch prediction. In *ICPRAM*, pages 520–527, 2014.

[7] Erik L Heiny and David Blevins. Predicting the atlanta falcons play-calling using discriminant analysis. *Journal of Quantitative Analysis in Sports*, 7(3), 2011.

[8] Craig Joash Fernandes, Ronen Yakubov, Yuze Li, Amrit Kumar Prasad, and Timothy CY Chan. Predicting plays in the national football league. *Journal of Sports Analytics*, 6(1):35–43, 2020.

[9] Namhoon Lee and Kris M Kitani. Predicting wide receiver trajectories in american football. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016.

[10] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

[11] Todd G Nick and Kathleen M Campbell. Logistic regression. *Topics in biostatistics*, pages 273–301, 2007.

[12] Marius Ötting. Predicting play calls in the national football league using hidden markov models. *IMA Journal of Management Mathematics*, 32(4):535–545, 2021.

[13] Samet Oymak and Mahdi Soltanolkotabi. Toward moderate overparameterization: Global convergence guarantees for training shallow neural networks. *IEEE Journal on Selected Areas in Information Theory*, 1(1):84–105, 2020.

[14] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1):1934–1965, 2019.

[15] K Srujan Raju, M Ramakrishna Murty, M Varaprasad Rao, and Suresh C Satapathy. Support vector machine with k-fold cross validation model for software fault prediction. *International Journal of Pure and Applied Mathematics*, 118(20):321–334, 2018.

[16] Derek D Reed, Thomas S Critchfield, and Brian K Martens. The generalized matching law in elite sport competition: Football play calling as operant choice. *Journal of Applied Behavior Analysis*, 39(3):281–297, 2006.

[17] Cedric Seger. An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing, 2018.

[18] M Paz Sesmero, Agapito I Ledezma, and Araceli Sanchis. Generating ensembles of heterogeneous classifiers using stacked generalization. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 5(1):21–34, 2015.

[19] Jan N Van Rijn and Frank Hutter. Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2367–2376, 2018.

[20] Sascha Wilkens. Sports prediction and betting models in the machine learning age: The case of tennis. *Journal of Sports Analytics*, 7(2):99–117, 2021.

[21] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.